IFW

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	**Philippe Lafon**	Docket:	**TI-37335**
Serial No.:	**10/823,183**	Art Unit:	**TBD**
Filed:	**04/13/2004**	Examiner:	**TBD**
For:	**Method And Related System, For Overlaying A Graphics Object On A Digital Picture**	Conf. No.:	**TBD**

TRANSMITTAL LETTER ACCOMPANYING CERTIFIED COPY OF PRIORITY APPLICATION UNDER 35 U.S.C. § 119

MAILING CERTIFICATE UNDER 37 C.F.R. § 1.8(a)

I hereby certify that on this date, this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to the: Commissioner for Patents, Alexandria, VA 22313-1450 on May 26, 2004.

Elizabeth Austin

Commissioner for Patents
Alexandria, VA 22313-1450

Dear Sir:

Submitted herewith is a certified copy of European Patent Application No. 03293277.4, filed on 19/12/2003, in the European Patent Office and from which priority under 35 U.S.C. § (e)(1) is claimed for the above-identified application.

Please charge any fees necessary to the deposit account of Texas Instruments Incorporated, Account No. 20-0668. An original and two copies of this sheet are enclosed.

Respectfully submitted,

Ronald O. Neerings
Patent Attorney
Reg. No. 34,227

Texas Instruments Incorporated
Mail Station 3999
P.O. Box 655474
Dallas, TX 75265
(972) 917-5299

Enclosed: Certified Copy of Priority Document

Europäisches Patentamt | European Patent Office | Office européen des brevets

Bescheinigung | Certificate | Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. | Patent application No. | Demande de brevet n°

03293277.4

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

R C van Dijk

Anmeldung Nr:
Application no.: 03293277.4
Demande no:

Anmeldetag:
Date of filing: 19.12.03
Date de dépôt:

Anmelder/Applicant(s)/Demandeur(s):

TEXAS INSTRUMENTS INCORPORATED
P.O. Box 655474,
13500 Central Expressway
Dallas, TX 75265
ETATS-UNIS D'AMERIQUE

Texas Instruments France
Avenue Bel Air BP5
06271 Villeneuve Loubet Cedex,
Nice
FRANCE

Bezeichnung der Erfindung/Title of the invention/Titre de l'invention:
(Falls die Bezeichnung der Erfindung nicht angegeben ist, siehe Beschreibung.
If no title is shown please refer to the description.
Si aucun titre n'est indiqué se referer à la description.)

Video Overlay

In Anspruch genommene Prioriät(en) / Priority(ies) claimed /Priorité(s) revendiquée(s)
Staat/Tag/Aktenzeichen/State/Date/File no./Pays/Date/Numéro de dépôt:

Internationale Patentklassifikation/International Patent Classification/
Classification internationale des brevets:

H04N5/272

Am Anmeldetag benannte Vertragstaaten/Contracting states designated at date of filing/Etats contractants désignées lors du dépôt:

AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LU MC NL
PT RO SE SI SK TR LI

THIS PAGE BLANK (USPTO)

VIDEO OVERLAY

Special effects such as graphic overlaying allow inserting graphics objects, such as borders, titles and dates, into the pictures or the real-time video.
The color-key overlaying method is based on a pre-defined "color-key" value which is used to indicate when a graphic pixel is transparent. When it is NOT set to the color key value, it is NOT transparent, so it must replace the current pixel of the input picture. Because the overlaying needs to be performed very accurately, the switching decision occurs for each pixel of the picture, in the 4:4:4 space (3 components per pixel). Therefore the graphic overlay is created in the 4:4:4 format (YUV or RGB). Usually, video or pictures are present in the 4:2:0 format in systems; therefore, devices need to convert them into the 4:4:4 space in order to perform the overlaying operation. Reverse conversion is also required since 4:2:0 format is used for the encoding.

- The disclosed device provides a (possibly non real time) pre-processing method to transform the graphic overlay into the Y,Cb,Cr 4:2:0 space, with some specific weight calculation applied to the chrominance (Cb,Cr) components. To help real time operations, those pre-calculated weight values may optionally be embedded or attached to this resulting 4:2:0 graphic overlay.
- It provides the associated switching method to be then applied (in real time) to the input picture or the video stream directly in 4:2:0 space: Pixel switching on the Luminance (Y) and weighted switching on the chrominance (Cb,Cr).
- The overall system is presented using a simple interpolation method however, other types of interpolations methods may also be defined using the same approach.
- Finally the device can also be extended to alpha blending overlaying.

- 4:2:0 to 4:4:4 and, 4:4:4 to 4:2:0 real time transforms, as well as YUV to RGB color conversions may be removed, with a large computational saving.
- The luminance switching quality is unchanged since the luminance is not re-sampled.
- The chrominance switching quality is weighted in order to offer the equivalent quality as devices using formats conversions.

ABSTRACT:
The purpose of this invention is to provide a simple and efficient way to perform real time color key switching directly in the 4:2:0 Y,Cb,Cr space, with a specific associated preprocessing stage to transform the usual color key graphic overlay from the 4:4:4 space into the 4:2:0 space. This preprocessing stage is not real time constrained since the graphic overlay is selected by the user, and can be prepared in a reasonably long period of time.
Because the overlay switching is done in the 4:2:0 space, a weighted switching is applied to the chrominance samples, while usual pixel switching is applied to the luminance samples.

1. Introduction:

New generation Multimedia wireless products now offer picture and video capture and compressions capabilities. Special effects such as graphic overlaying allow inserting graphics objects, such as borders, titles and dates, into the pictures or the real-time video.

Usually picture or video compression engines use internally the Y,Cb,Cr 4:2:0 format, while the overlay color-key switching is defined in the RGB 4:4:4 space, and needs to be operated in this space, since it is a pixel switching based method.
For this reason the incoming video needs to be converted into the 4:4:4 space (optionally by applying a YUV to RGB matrix), then the overlaying process is applied. Finally the reverse transforms are required, prior to using the encoder for compressing the signal.

Extra cost brought by these color space and formats conversions have motivated the study of this novel method which combines both pixel based switching and weighted (alpha-blending) switching. The pixel based switching is applied to the luminance, while the weighted switching is applied to the chrominance components.

Finally a large computation saving is reached since the real time format conversions are removed and the switching is applied to a more reduced number of samples defined by the 4:2:0 space. For this last reason, the memory requirement is also reduced.

2. *General concept:*

2.1. *Overlay and color key definition in 4:4:4 space:*

- Let's define the graphic overlay as a picture **OVLY444** where a color level (here Y,Cb,Cr) 4:4:4 is reserved to define if a pixel is transparent or not. This level is called CK (Color Key).

Example: CK = [0(y), 0(cb), 0(cr)]

- Let's define the input picture to be **PIC444**, with the same size in the same 4:4:4 space than the OVLY picture.

We can define the normal overlay process as the following:

For each PIC444[Y,Cb,Cr] input sample, if the corresponding OVLY444[Y,Cb,Cr] sample has the value CK, the picture sample is maintained, otherwise, it is replaced by the OVLY444[Y,Cb,Cr] sample.

Algorithm:

```
For Each input picture pixels (i,j) {
    if ( OVLY444(i,j)[Y,Cb,Cr] != CK ) {
        PIC444(i,j)[Y,Cb,Cr] = OVLY444(i,j)[Y,Cb,Cr];
    }
}
```

With: j varying from 0 to picture width and i varying from 0 and picture height.

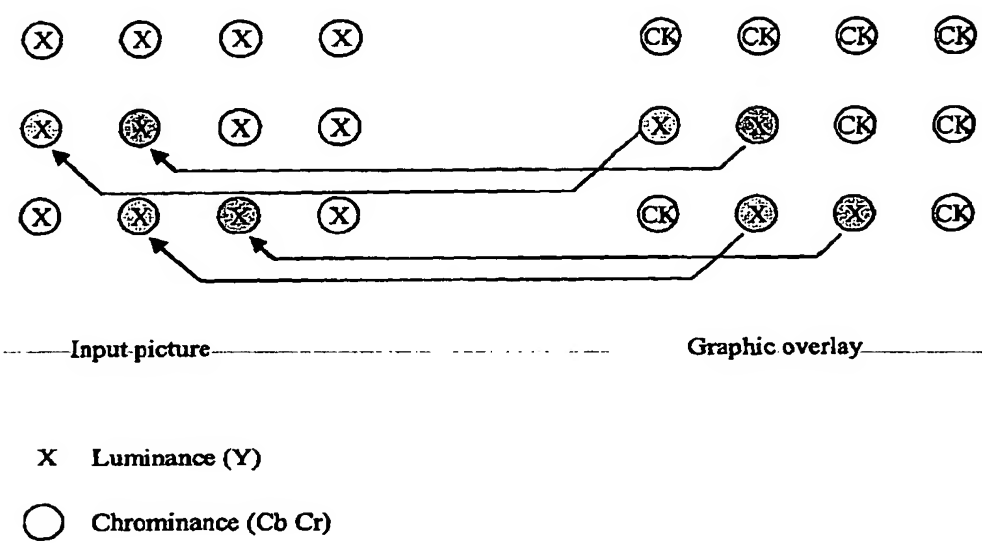

Figure 1: Pixel switching mechanism

2.2. Format representations

The 4:4:4 to 4:2:0 conversion can be viewed as a linear or bilinear transform that doesn't affect the luminance samples, and that divides by two the resolution of the chrominance samples in both the horizontal and the vertical directions.

We represent here the spatial location of the chrominance samples associated to their luminance samples in both the 4:4:4 space and the 4:2:0 space:

- In the 4:4:4 space the chrominance samples apply directly to the same luminance samples location.
- In the 4:2:0 space the chrominance samples apply to surrounding luminance samples and are defined in location where contributions to the top and bottom are equal. Note that other type of 4:2:0 formats may be used based on other phase location for the chrominance.

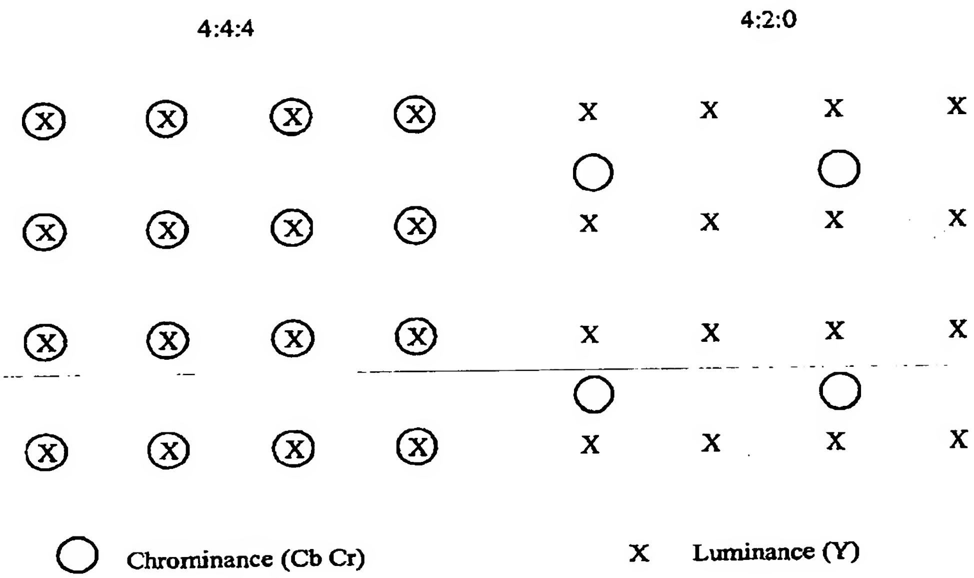

Figure 2: 4:4:4 and 4:2:0 representations

2.3. Simplified 4:4:4 to 4:2:0 conversion:

We define here a simple transform from 4:4:4 to 4:2:0. This transform has been chosen because it maintains a very acceptable quality while the complexity is not too high. In addition the chrominance samples only depend on the adjacent surrounding pixels.

$$T = (A/8 + B/4 + C/8) + (D/8 + E/4 + F/8) \quad \text{(see fig. 3)}$$

Where: T is the resulting chrominance sample in the 4:2:0 space
And: A,B,C,D,E,F are surrounding chrominance samples from the 4:4:4 space.

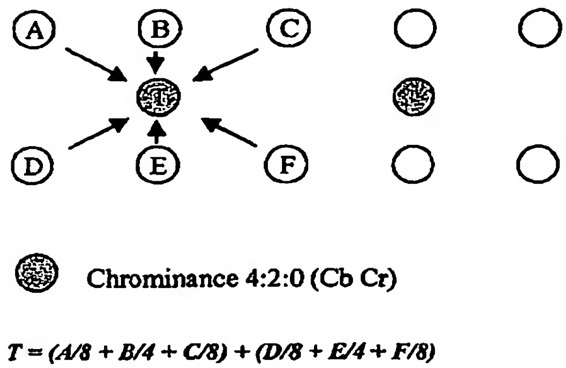

Figure 3: Chrominance 4:4:4 to 4:2:0

Note:

This simplified conversion **cannot** be applied directly to the color key graphic overlay since some overlay pixels may or may not have a valid contribution into the 4:2:0 space.

3. Modifying the graphic Overlay into the 4:2:0 space:

The following method is simply based on the use of the 4:4:4 to 4:2:0 chrominance interpolation, however, only the non transparent chrominance samples must contribute to the interpolation.

The process can be separated into the luminance process where the 4:2:0 and 4:4:4 spaces do not differ, and the chrominance process where the 4:2:0 and 4:4:4 space are different.

We define several rules:

1- OVLY luminance (Y) level **0** is reserved for "***transparency***", levels [1-255] are the active levels containing the luminance value to replace the respective luminance sample in the 4:2:0 picture.

2- When a Y pixel is ***active*** [1-255] its associated chrominance (Cb,Cr) levels have an active contribution level that "replaces", part of the chrominance sample of the 4:2:0 picture, proportionally to a "***weight***".

3- When a Y pixel is "***transparent***" the contribution of its associated (Cb,Cr) samples is zero.

4- The total "weight"of contribution of the (Cb,Cr) 4:4:4 samples into the 4:2:0 (Cb,Cr) sample can be defined by the sum of the "weights" of those of the active contributors. For example each individual weight is 1/8 or 1/4 while the total weigh <= 1. (based on T = (A/8 + B/4 + C/8) + (D/8 + E/4 + F/8)).

3.1. Overlay transform (non real time preprocessing):

Based on the above definitions and rules we can now define the transform of the graphic overlay itself:

- Let's define the 4:2:0 target overlay graphic as OVLY420.

Luminance:

Algorithm:

```
For Each OVLY420(i,j) {
    if ( OVLY444(i,j) == CK ) {                           //transparent
        OVLY420(i,j)[Y] = 0;
    }
    else {                                                //active
        OVLY420(i,j)[Y] = OVLY444(i,j)[Y];                //copy Y
        If (OVLY420(i,j)[Y] == 0)                         //avoid zero
            OVLY420(i,j)[Y] = 1;                          //almost black
    }
}
```

With: j varying from 0 to picture width and i varying from 0 and picture height.

For the luminance we simply ensure that when a Y sample is transparent, Y is set to zero, otherwise it must not be a zero value.

Chrominance:

Algorithm:

```
For Each OVLY420(k,l) {
    OVLY420(k,l)[Cb,Cr]  = 0;                             //Cb Cr
    //A contributor
    if ( OVLY444(2k  ,2l-1) != CK )                       //active
        OVLY420(k,l)[Cb,Cr] += OVLY444(2k  ,2l-1)[Cb,Cr] / 8;
    //B contributor
    if ( OVLY444(2k  ,2l  ) != CK )                       //active
        OVLY420(k,l)[Cb,Cr] += OVLY444(2k  ,2l  )[Cb,Cr] / 4;
    //C contributor
    if ( OVLY444(2k  ,2l+1) != CK )                       //active
        OVLY420(k,l)[Cb,Cr] += OVLY444(2k  ,2l+1)[Cb,Cr] / 8;
    //D contributor
    if ( OVLY444(2k+1,2l-1) != CK )                       //active
        OVLY420(k,l)[Cb,Cr] += OVLY444(2k+1,2l-1)[Cb,Cr] / 8;
    //E contributor
    if ( OVLY444(2k+1,2l  ) != CK )                       //active
        OVLY420(k,l)[Cb,Cr] += OVLY444(2k+1,2l  )[Cb,Cr] / 4;
    //F contributor
    if ( OVLY444(2k+1,2l+1) != CK )                       //active
        OVLY420(k,l)[Cb,Cr] += OVLY444(2k+1,2l+1)[Cb,Cr] / 8;
}
```

With: l varying from 0 to picture width/2 and k varying from 0 and picture height/2.

In our example below we obtain:

T = 0 A/8 + 1 B/4 + 1 C/8 + 1 D/8 + 0 E/4 + 0 F/8

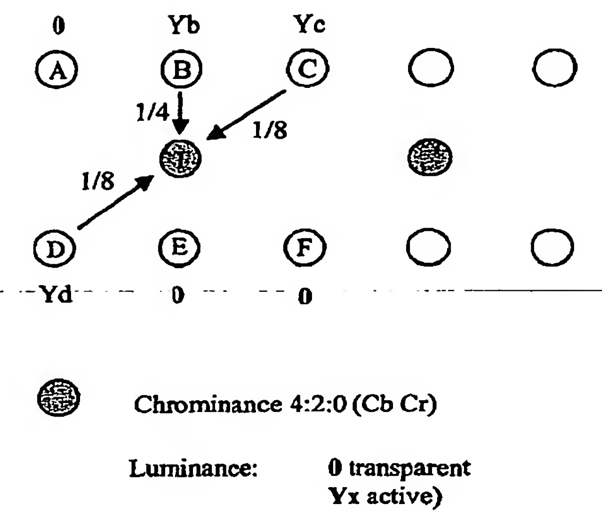

Figure 4: Chrominance switch and weighted contributions 4:4:4 to 4:2:0

Picture bordering:

When A an D are beyond the left border position (line beginning) they do not exist, instead we can set A = B, and D = E. thus the overall weighting definition is maintained. The same rule applies to the associated weight calculation.

3.2. Associated Weight:

For each calculated chrominance 4:2:0 samples, from the 4:4:4 overlay graphic, a weight (W) can be calculated: This is simply done by adding each individual weight when its respective luminance sample is active. This weight value is used at overlaying level to perform an alpha blending operation as shown in the overlaying process.

In our chosen example we obtain:

W(T) = 0 1/8 + 1 1/4 + 1 1/8 + 1 1/8 + 0 1/4 + 0 1/8
W(T) = 1/2

Note:

The weighting value may be embedded within the chrominance value if the chrominance number of levels is limited, such that enough free bits are left available for it. In our example, 4 bits are required to store the weight. Otherwise the transparency level can also be associated with each chrominance sample, but this requires more data storage. Finally the weight can also be reconstructed during the overlay switching operation, at the cost of real time operations.

3.3. *4:2:0 overlaying (real time process):*

Let's define **PIC420** the 4:2:0 input picture with the same size and format than the OVLY420 overlay graphic.

It is now possible to do the luminance switch, and to do a weighted chrominance switch, since each Y from OVLY420 contains either the Y active values or inactive status. Thus the chrominance weight reconstruction simply uses the surrounding Y active status, finally Cb,Cr contains the interpolated contribution of OVLY444 to be weighted with the Cb,Cr samples of the picture.

Luminance:

Algorithm:

```
For Each PIC420(i,j) {
    if ( OVLY420(i,j)[Y] != 0 )                    //active
        PIC420(i,j)[Y] = OVLY420(i,j)[Y];
}
```

The luminance process is trivial since it is just replacing Y samples of the picture when Y samples of the overlay are not zero.

Chrominance:

Algorithm:

```
For Each PIC420(k,l) {

    /*** Weight reconstruction */
    W = 0;                                         // No weigth
    //A contributor
    if ( OVLY420(2k  ,2l-1)[Y] != 0 )              //active
        W += 1;
    //B contributor
    if ( OVLY420(2k  ,2l  )[Y] != 0 )              //active
        W += 2;
    //C contributor
    if ( OVLY420(2k  ,2l+1)[Y] != 0 )              //active
        W += 1;
    //D contributor
    if ( OVLY420(2k+1,2l-1)[Y] != 0 )              //active
        W += 1;
    //E contributor
    if ( OVLY420(2k+1,2l  )[Y] != 0 )              //active
        W += 2;
    //F contributor
    if ( OVLY420(2k+1,2l+1)[Y] != 0 )              //active
        W += 1;

    /*** alpha blending */
    PIC420(k,l)[Cb,Cr] =
          PIC420(k,l)[Cb,Cr] * (8 - W) / 8
        + OVLY420(k,l)[Cb,Cr];  // x W/8 done in pre-processing
}
```

Here the largest part of the calculation is to reconstruct the total weight, however, as stated earlier, the weighting may be provided attached to (or within) the OVLY420 graphic overlay, and may not need to be reconstructed.
Finally, the total weight allows switching proportionally (weighted) contributions from the graphic overlay (W) replacing with the same amount from the input picture sample contribution (1-W).

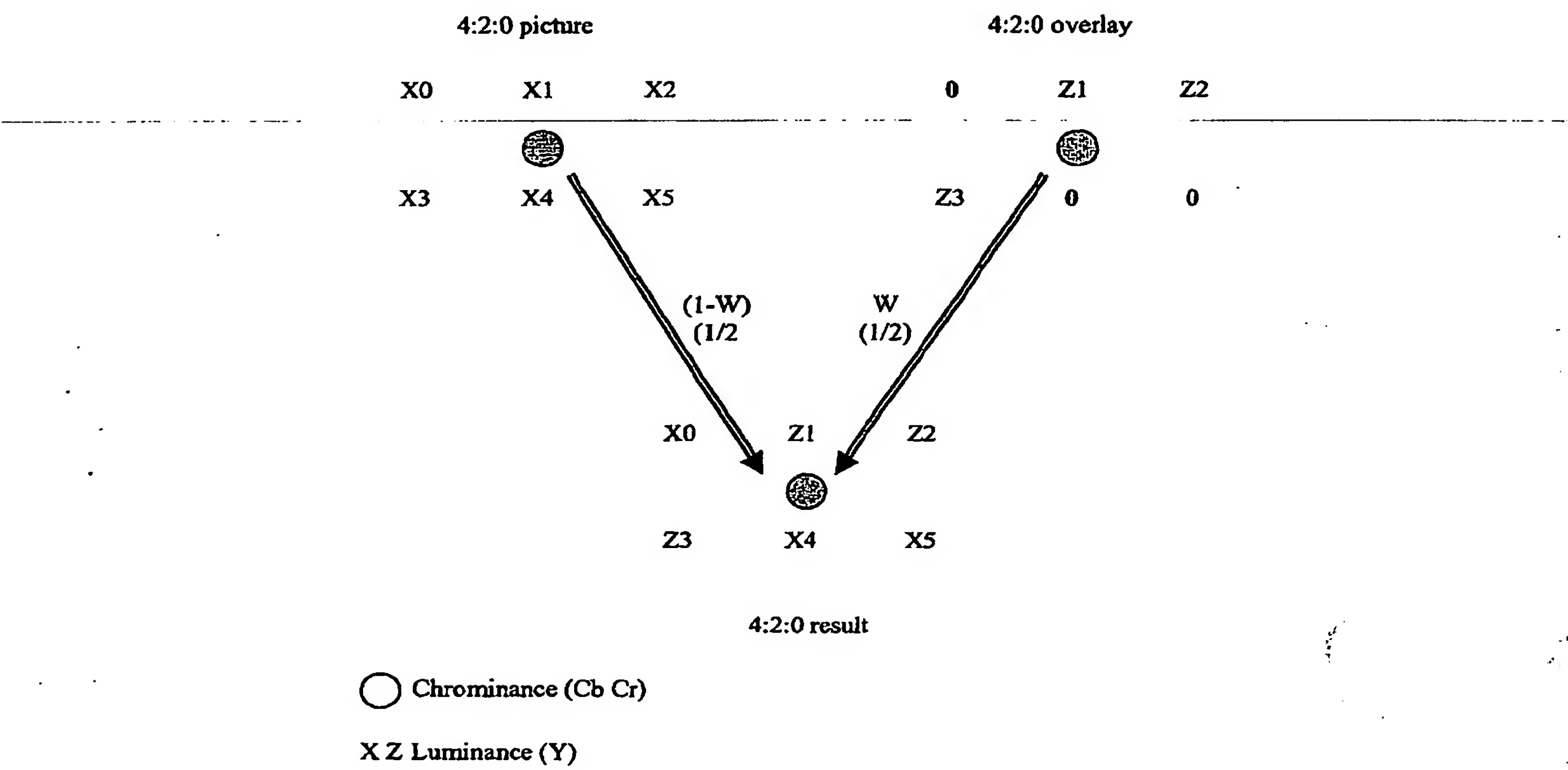

Figure 5: Luminance switch / Chrominance weighted switch

4. *Extension to the alpha blending overlaying:*

The domain of application of this invention can be extended very directly to alpha blending overlaying:

Using the switch case:

$$\mathbf{W(T) = 0{,}1/8 + 0{,}1/4 + 0{,}1/8 + 0{,}1/8 + 0{,}1/4 + 0{,}1/8}$$

We can extend it to:

$$\mathbf{W\alpha(T) = \alpha a/8 + \alpha b/4 + \alpha c/8 + \alpha d/8 + \alpha e/4 + \alpha f/8}$$

Where 0 <= α <= 1 represents the opacity value of each luminance pixel of the graphic overlay.

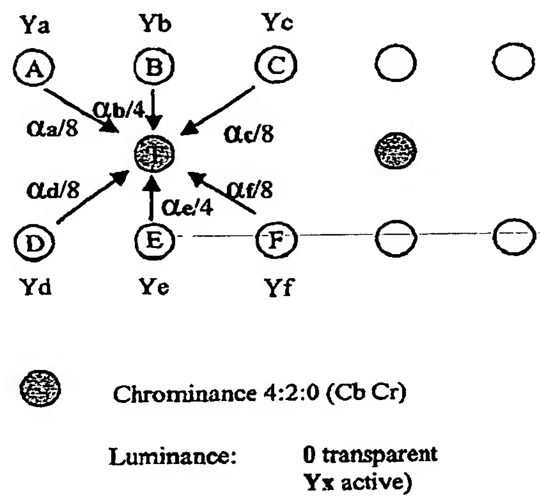

Figure 6: Chrominance Alpha weighted contributions 4:4:4 to 4:2:0

In our example we obtain for T:

$$\mathbf{T = A\ \alpha a/8 + B\ \alpha b/4 + C\ \alpha c/8 + D\ \alpha d/8 + E\ \alpha e/4 + F\ \alpha f/8}$$
$$\mathbf{W\alpha(T) = \alpha a/8 + \alpha b/4 + \alpha c/8 + \alpha d/8 + \alpha e/4 + \alpha f/8}$$

4.1. Overlay transform (non real time preprocessing):

In the same manner the graphic is transformed into 4:2:0 by using individual **α** values instead of 0 or 1:

Luminance:

For the luminance we ***pre-multiply*** the samples by the associated α values.

Algorithm:

```
/*** prepare luminance in advance α-multiplication */
For Each OVLY420(i,j) {
    OVLY420(i,j)[Y] = OVLY(i,j)[Y] * α(i,j);
}
```

Chrominance:

For the chrominance we interpolate the chrominance sample and the weighting value:Algorithm:

```
For Each OVLY420(k,l) {

    /*** chrominance sample */
    OVLY420(k,l)[Cb,Cr] =
    //A contributor
            OVLY(2k  ,2l-1)[Cb,Cr] * α(2k  ,2l-1) / 8
    //B contributor
          + OVLY(2k  ,2l  )[Cb,Cr] * α(2k  ,2l  ) / 4
    //C contributor
          + OVLY(2k  ,2l+1)[Cb,Cr] * α(2k  ,2l+1) / 8
    //D contributor
          + OVLY(2k+1,2l-1)[Cb,Cr] * α(2k+1,2l-1) / 8
    //E contributor
          + OVLY(2k+1,2l  )[Cb,Cr] * α(2k+1,2l  ) / 4
    //F contributor
          + OVLY(2k+1,2l+1)[Cb,Cr] * α(2k+1,2l+1) / 8;

    /*** global weight */
    Wα(k,l) =
    //A contributor
            α(2k  ,2l-1) / 8
    //B contributor
          + α(2k  ,2l  ) / 4
    //C contributor
          + α(2k  ,2l+1) / 8
    //D contributor
          + α(2k+1,2l-1) / 8
    //E contributor
          + α(2k+1,2l  ) / 4
    //F contributor
          + α(2k+1,2l+1) / 8;
}
```

4.2. 4:2:0 overlaying (real time process):

The rest of the process uses normal alpha blending for the luminance, as in 4:4:4.

Luminance:

Algorithm:

```
For Each PIC420(i,j) {

    /*** alpha blending */
    PIC420(i,j)[Y] =
        (1 - α(i,j)) PIC420(i,j)[Y]
      + OVLY420(i,j)[Y];    // x α(i,j) done in pre-processing
};
```

Chrominance:

For the chrominance part it uses also alpha blending (as in the color key overlay), with alpha being the **Wα** value replacing the W weighting value from the color key device.

Algorithm:

```
For Each PIC420(k,l) {
    /*** alpha blending ***/
    PIC420(k,l)[Cb,Cr] =
        PIC420(k,l)[Cb,Cr] * (1 - wα(k,l))
        + OVLY420(k,l)[Cb,Cr]; // x wα(k,l) done in pre-processing
}
```

Note:

For both the luminance and the chrominance the α and Wα values will be placed into an array attached to the 4:2:0 graphic overlay, so that the real time process can access to them.

This invention offers an original and very efficient way to overlay graphics/colour-key onto pictures or real time video flows. For both speed and architectural reasons, this method was invented to allow overlay into the YUV 4:2:0 space instead of 4:4:4.

Implementing color-key based switching overlaying on DSP requires a lot of operations and is defined in the RGB 4:4:4 space. The result must then be transformed into the overlaying directly into the YUV 4:2:0 space before it is encoded (compressed). The proposed solution allows to do the overlaying directly into the YUV 4:2:0 space and saves YUV 4:2:0 to YUV 4:4:4 and then YUV 4:4:4 to YUV 4:2:0 transform stages. Thus the largest part of cycles can be saved into systems with embedded processors such as DSP.

Usual known solutions perform the pixel overlaying (switching), into the RGB or YUV 4:4:4 space. Switching must be done on each component. Then 4:4:4 to 4:2:0 transform is required when coming back to the 4:2:0 usual encoding format.

The proposed solution provides a specific method to preprocess the graphic YUV 4:4:4 color-key pictures into a YUV 4:2:0 space (non real time constrained) with some additional information embedded called weighting. Then specific 4:2:0 space switching is provided. (real time constrained operation). This stage contains minimized calculations or operations.

The solution is simple in a sense that is saved many MIPS/calculation expensive transform stages, without compromising the visual quality of the result. Also memory buffering requirements are reduced, since there is no need to work on the 4:4:4 space (half space required in 4:2:0).

CLAIMS:

1. A preprocessing method for graphic data to transform a graphic overlay into a transform space.

2. the material of this description in any novel or inventive combination.